Introducción a las Redes Neuronales

Cosijopii

January 17, 2025

Temario

¿Qué son las Redes Neuronales?

- Modelo computacional inspirado en el cerebro humano.
- Compuesto por neuronas artificiales conectadas entre sí.
- Cada conexión tiene un peso que ajusta su influencia en el aprendizaje.

Ejemplo: Una red neuronal que reconoce dígitos escritos a mano.

Historia de las Redes Neuronales

- **1950-1960:** Desarrollo del perceptrón (Frank Rosenblatt).
- **1980:** Introducción del algoritmo de *Backpropagation*.
- **2000 en adelante:** Avances en hardware y deep learning.

Hitos recientes:

- Modelos como GPT, DALL-E, etc.
- Aplicaciones en visión por computadora y procesamiento de lenguaje natural.

Importancia y Aplicaciones

Aplicaciones más comunes:

- Reconocimiento de imágenes y voz.
- Predicción de datos (series temporales, clima, mercados financieros).
- Diagnóstico médico (detección de enfermedades).
- Automatización (vehículos autónomos, chatbots).

Impacto: Solución de problemas complejos que antes eran imposibles de abordar.

Modelo de Neurona y Regla de Hebb

Modelo de una Neurona:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Donde:

- x_i: Entrada i.
- w_i: Peso asociado a x_i.
- b: Sesgo.
- f: Función de activación (por simplicidad usaremos $f(x) = \text{step}(x)$).

Regla de Hebb:

$$\Delta w_{ij} = \eta \cdot x_i \cdot y_j$$

Ajusta los pesos w_{ij} reforzando conexiones activadas simultáneamente.

Ejemplo Ampliado: Actualización de Pesos

Paso 1: Definir los datos iniciales.

- Tasa de aprendizaje: $\eta = 0.1$.
- Pesos iniciales: $w = [0.0, 0.0]$ (se inicializan en 0).
- Sesgo: $b = 0$ (sin sesgo para simplificar).
- Puntos de entrenamiento:

$$\text{Entradas: } \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \text{Salida esperada: } y$$

Punto	Entrada $[x_1, x_2]$	Salida esperada y
1	$[1, 1]$	1
2	$[1, 0]$	0
3	$[0, 1]$	0

Objetivo: Aprender a clasificar correctamente los puntos usando la Regla de Hebb.

Ejemplo Ampliado: Paso 2 - Actualización de Pesos I

Cálculo de pesos iterando por los puntos:

- Inicialización:

$$w = [w_1, w_2] = [0.0, 0.0]$$

- Actualización para cada punto:

Punto 1: $[x_1, x_2] = [1, 1],\ y = 1$

$$\Delta w_1 = \eta \cdot x_1 \cdot y = 0.1 \cdot 1 \cdot 1 = 0.1$$

$$\Delta w_2 = \eta \cdot x_2 \cdot y = 0.1 \cdot 1 \cdot 1 = 0.1$$

Nuevos pesos:

$$w = [0.0 + 0.1, 0.0 + 0.1] = [0.1, 0.1]$$

Punto 2: $[x_1, x_2] = [1, 0],\ y = 0$

$$\Delta w_1 = \eta \cdot x_1 \cdot y = 0.1 \cdot 1 \cdot 0 = 0$$

Ejemplo Ampliado: Paso 2 - Actualización de Pesos II

$$\Delta w_2 = \eta \cdot x_2 \cdot y = 0.1 \cdot 0 \cdot 0 = 0$$

Pesos permanecen iguales:

$$w = [0.1, 0.1]$$

Punto 3: $[x_1, x_2] = [0, 1]$, $y = 0$

$$\Delta w_1 = \eta \cdot x_1 \cdot y = 0.1 \cdot 0 \cdot 0 = 0$$

$$\Delta w_2 = \eta \cdot x_2 \cdot y = 0.1 \cdot 1 \cdot 0 = 0$$

Pesos finales:

$$w = [0.1, 0.1]$$

Ejemplo Ampliado: Paso 3 - Evaluación de la Neurona

Evaluación con un nuevo punto:

- Nuevo punto: $[x_1, x_2] = [0, 0]$.
- Función de activación:

$$f(x) = \text{step}(x) = \begin{cases} 1, & \text{si } x \geq 0 \\ 0, & \text{si } x < 0 \end{cases}$$

- Cálculo de la salida:

$$\text{Potencial neto: } \text{net} = w_1 \cdot x_1 + w_2 \cdot x_2 = 0.1 \cdot 0 + 0.1 \cdot 0 = 0$$

$$\text{Salida: } y = f(\text{net}) = f(0) = 0$$

Conclusión: La neurona predice correctamente que el punto $[0, 0]$ pertenece a la clase 0.

Ejercicio para Resolver a Mano

Problema:

- Pesos iniciales: $w = [0.0, 0.0]$.
- Tasa de aprendizaje: $\eta = 0.2$.
- Datos:

Punto	Entrada $[x_1, x_2]$	Salida esperada y
1	$[1, 0]$	1
2	$[0, 1]$	1
3	$[0, 0]$	0

Tareas:

1. Actualice los pesos w_1, w_2 para cada punto.
2. Determine los pesos finales.
3. Evalúe la salida para el nuevo punto $[1, 1]$.

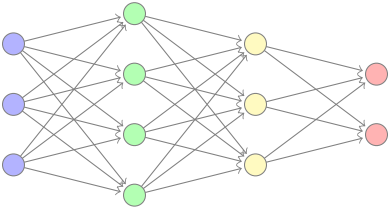
Solución:

Componentes de una Red Neuronal

Estructura de una Red Neuronal Artificial:

- **Capas:** Entrada, ocultas y salida.
- **Neuronas:** Unidades de procesamiento.
- **Conexiones:** Pesos entre neuronas.
- **Flujo de información:** Propagación hacia adelante (*feedforward*).

Ejemplo de una red con dos capas ocultas:



Funciones de Activación

Definición: Introducen no linealidad en las redes neuronales, permitiendo a la red modelar problemas más complejos.

Ejemplos comunes:

- **Sigmoide:**

$$f(x) = \frac{1}{1 + e^{-x}}$$

  Rango: $[0, 1]$.

- **ReLU (Unidad Lineal Rectificada):**

$$f(x) = \max(0, x)$$

- **Tanh:**

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

  Rango: $[-1, 1]$.

Gráfico comparativo:

Redes Feedforward

Definición: Redes neuronales donde la información fluye en una sola dirección, desde la capa de entrada hasta la capa de salida, sin ciclos.

Estructura típica:

- Capas completamente conectadas.
- Uso de funciones de activación en cada neurona.

Aplicaciones comunes:

- Clasificación.
- Regresión.
- Predicción.

Limitaciones:

- No pueden manejar datos secuenciales o temporales.

Ejemplo de una Red Feedforward I

Ejemplo de flujo de información:

- Entrada: $[x_1, x_2]$.
- Pesos: $W_1 = [0.5, -0.3]$, $W_2 = [0.8, 0.1]$.
- Función de activación: ReLU.

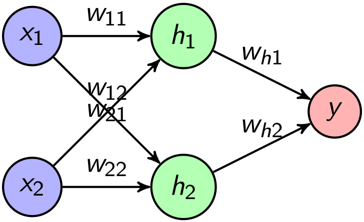
Cálculo:

$$\text{Capa oculta: } h_1 = \max(0, W_1 \cdot x), \quad h_2 = \max(0, W_2 \cdot x)$$

$$\text{Salida: } y = W_{\text{salida}} \cdot h$$

Diagrama:

Ejemplo de una Red Feedforward II



Backpropagation: Introducción

¿Qué es Backpropagation?

- Algoritmo para entrenar redes neuronales.
- Utiliza el **descenso de gradiente** para ajustar los pesos y minimizar el error.

Fases del algoritmo:

1. Propagación hacia adelante (*forward pass*): Calcula la salida de la red.
2. Propagación hacia atrás (*backward pass*): Calcula los gradientes del error respecto a los pesos utilizando la regla de la cadena.
3. Actualización de pesos: Ajusta los pesos usando los gradientes calculados.

Función de Pérdida L

- La función de pérdida mide el error entre la predicción de la red ($\hat{y}$) y el valor real (y).
- Ejemplo: **Error Cuadrático Medio (MSE)**:

$$L = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

- Otra opción común: **Entropía Cruzada (Cross-Entropy)**:

$$L = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

- El objetivo es minimizar L ajustando W y b.

El Papel de las Derivadas Parciales

- Para minimizar L, necesitamos calcular $\frac{\partial L}{\partial W}$ y $\frac{\partial L}{\partial b}$ para cada capa de la red.
- Esto nos dice cómo cambia L cuando ajustamos ligeramente W o b.
- Usamos el **gradiente descendente**:

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}, \quad b \leftarrow b - \eta \frac{\partial L}{\partial b}$$

donde η es la tasa de aprendizaje.

Teorema de la Cadena en Backpropagation

- El error L depende indirectamente de los parámetros de las capas ocultas.
- Usamos la regla de la cadena:

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial W}$$

- Donde:
  - $\hat{y}$: Salida de la red.
  - $z = W \cdot x + b$: Entrada ponderada.

Cálculo de las Derivadas

1. **Derivada de la pérdida respecto a la salida:**

$$\frac{\partial L}{\partial \hat{y}}$$

Ejemplo:

$$\text{Para MSE: } \frac{\partial L}{\partial \hat{y}} = 2(\hat{y} - y)$$

2. **Derivada de la salida respecto a la entrada ponderada:**

$$\frac{\partial \hat{y}}{\partial z} = f'(z)$$

Donde $f(z)$ es la función de activación. Ejemplos:

- Sigmoide: $f'(z) = f(z)(1 - f(z))$
- ReLU: $f'(z) = 1 \text{ si } z > 0, \text{ sino } 0$

3. **Derivada de z respecto a los parámetros:**

$$\frac{\partial z}{\partial W} = x, \quad \frac{\partial z}{\partial b} = 1$$

Producto Elemento a Elemento ($\odot$)

- El **producto elemento a elemento** (*Hadamard product*) se denota por $\odot$.
- Es una operación entre dos vectores u y v del mismo tamaño:

$$(u \odot v)_i = u_i \cdot v_i$$

- Ejemplo:

$$u = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \quad v = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} \quad \Rightarrow \quad u \odot v = \begin{bmatrix} 4 \\ 10 \\ 18 \end{bmatrix}$$

- En backpropagation, se usa para combinar gradientes:

$$\frac{\partial L}{\partial z^{(l)}} = \frac{\partial L}{\partial a^{(l)}} \odot f'(z^{(l)})$$

Resumen del Proceso de Backpropagation

1. **Inicio:** Calcula el error en la capa de salida.

$$\frac{\partial L}{\partial \hat{y}}$$

2. **Propagación hacia atrás:** Usa la regla de la cadena para calcular gradientes en cada capa:

$$\frac{\partial L}{\partial z^{(l)}} = \left(W^{(l+1)}\right)^T \frac{\partial L}{\partial z^{(l+1)}} \odot f'(z^{(l)})$$

3. **Actualización de parámetros:**

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}, \quad b \leftarrow b - \eta \frac{\partial L}{\partial b}$$

Regla de la Cadena y Gradientes I

Concepto Clave: Regla de la Cadena

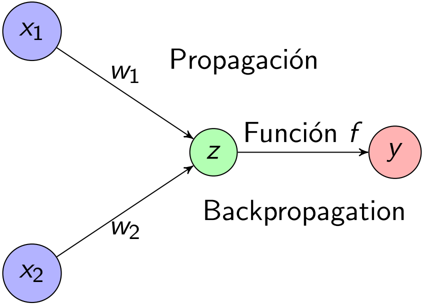
$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial w}$$

Donde:

- L: Función de pérdida.
- y: Salida de la neurona.
- z: Potencial neto de la neurona.
- w: Peso que conecta una neurona con la anterior.

Ejemplo visual:

Regla de la Cadena y Gradientes II



Actualización de Pesos I

Fórmula de Actualización de Pesos:

$$w^{(t+1)} = w^{(t)} - \eta \cdot \frac{\partial L}{\partial w}$$

Donde:

- η: Tasa de aprendizaje (learning rate).
- $\frac{\partial L}{\partial w}$: Gradiente de la pérdida respecto al peso.

Ejemplo numérico:

- Peso inicial: $w = 0.5$.
- Gradiente calculado: $\frac{\partial L}{\partial w} = 0.2$.
- Tasa de aprendizaje: $\eta = 0.1$.
- Nueva actualización:

$$w^{(t+1)} = 0.5 - 0.1 \cdot 0.2 = 0.48$$

Funciones de Pérdida I

Definición: Miden la diferencia entre la salida predicha y la salida esperada.

Ejemplos comunes:

- **Error Cuadrático Medio (MSE):**

$$L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- **Entropía Cruzada:**

$$L = -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right]$$

Descenso de Gradiente

Definición: Optimización iterativa que ajusta los pesos para minimizar la función de pérdida.

Fórmula:

$$w^{(t+1)} = w^{(t)} - \eta \cdot \nabla L$$

Variantes:

- **Descenso de Gradiente Estocástico (SGD):** Actualiza los pesos después de cada muestra.
- **Mini-batch SGD:** Usa pequeños lotes de datos para calcular gradientes.

Introducción a Adam

Algoritmo Adam:

- Combina el momento y el ajuste adaptativo de la tasa de aprendizaje.
- Calcula promedios móviles de los gradientes y sus cuadrados.

Ecuaciones:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$w^{(t+1)} = w^{(t)} - \eta \cdot \frac{m_t}{\sqrt{v_t} + \epsilon}$$

Ventajas:

- Rápida convergencia.
- Robusto para problemas con grandes dimensiones.